N91-10607

EVOLVING IMPACT OF ADA ON A PRODUCTION SOFTWARE ENVIRONMENT

F. McGarry (NASA/GSFC)
L. Esker (CSC)
K. Quimby (CSC)

1.0 BACKGROUND (Chart 1)

Since 1985, the Software Engineering Laboratory (SEL) has been studying the impact of Ada and Ada-related technologies on the software development of production projects within the Flight Dynamics Division (FDD) at NASA/GSFC. Until then, all software development projects had used FORTRAN as the primary implementation language. The Ada development work began with a pilot project and a research project that paralleled a production FORTRAN development project (References 1 and 2). After this initial Ada experience, several later production projects were developed in Ada. For each project, the SEL has collected such detailed information as resource data, error data, component information, methodology, and project characteristics, so that the SEL could study the evolution of the use of Ada itself and the actual characteristics of the Ada development process (Reference 3).

Analysis of the Ada projects has led personnel to document lessons learned during the development of Ada projects (References 4 through 7). These lessons have provided valuable insight into the impact of Ada, especially in the following areas:

1. The impact of Ada on the software development process, that is, the impact Ada has on such measures as productivity, reliability, and maintainability.

2. The impact of Ada over time, as shown by the differences between the first, second, and third Ada projects.

3. The use of Ada and Ada features as the development environment gains more experience in using Ada.

4. The timeframe for realizing the benefits of using Ada.

1.1 ADA PROJECTS STUDIED (Chart 2)

Ada use within the FDD began in January 1985 with the GRODY project. As part of the preparation for developing this system, personnel first participated in a practice Ada project by implementing an electronic mail system (EMS). These two projects actually represent a first Ada experience.

After the GRODY project was well under way, two new Ada simulator projects for the GOES satellite began. GOADA, the dynamics simulator, and GOESIM, the telemetry simulator, collectively represent a second major experience with Ada. They are considered second projects because (1) some team members had previous experience in developing systems in Ada and (2) these two projects could draw on lessons learned from GRODY. Not only were the staffing profiles of the two GOES simulator teams different from the GRODY team, but the two GOES teams began using additional software tools available within the DEC Ada development environment.

Late in 1987 and 1988, two more projects, UARSTELS and Build 4 of FDAS began; these projects represent a third distinct Ada experience. Currently, two more Ada projects are in their early stages: EUVEDSIM and EUVETELS, but these projects are very early in their lifecycles and are not yet available for study.

1.2 PROJECT STATUS AND CHARACTERISTICS (Chart 3)

All totaled, Ada has been used on eight projects in the flight dynamics area. Two projects (EMS and GRODY) are completed; three (GOADA, GOESIM, and FDAS) are well into system testing; and one (UARSTELS) is in the implementation phase. The other two projects (EUVEDSIM and EUVETELS) are in the early requirements analysis phase. These projects range from nearly 6K to 163K SLOC in size, where SLOC is total source lines of code including comments, blanks, newly developed code, and reused code. These projects have required or are expected to require from 4 to 36 months to complete and had from three to seven people working on them. Although GRODY lasted for 36 months, it should be noted that most personnel on this project did not work fulltime on its development. The small EMS project could have been completed by 2 or 3 people; but since it was part of the Ada training for the GRODY project, all GRODY developers participated in some part of the EMS project.

2.0 ADA EVOLUTION

2.1 TEAM EXPERIENCE AND DEVELOPMENT ENVIRONMENT (Chart 4)

Of the eight Ada projects currently under way, six projects have progressed far enough to be studied: EMS, GRODY, GOADA, GOESIM, FDAS, and UARSTELS. All six of the projects studied have been staffed with personnel with a similar level of software development experience, an average of 4 to 5 years. Except for UARSTELS, each project also had personnel with a similar level of experience in the application. To date, the SEL has not observed any impact due to differences in team experience between projects.

It is also too early to observe any differences in the effect of varied levels of Ada experience on project development. The number of people who are formally trained in Ada and/or the number of those who have been on previous Ada projects is still too small. Only the first Ada projects have been completed. Some personnel on those projects have contributed to current, ongoing projects; however, there are not enough people in the environment, even on the most recent Ada projects, to significantly change the ratio of experienced Ada personnel to those with no Ada experience.

The use of tools has evolved somewhat from the first Ada projects. The practice Ada project (EMS) had only rudimentary tools available (compiler, linker, editor). GRODY made use of the DEC symbolic debugger (SD), and the Configuration Management System (CMS). All subsequent Ada projects are using these tools as well as the Language Sensitive Editor (LSE). Project personnel have also developed some additional tools in house to create package bodies and templates for the associated subunits they need to develop.

2.2 SOFTWARE CHARACTERISTICS (Chart 5)

Traditionally, software size has been described in terms of the lines of code developed for the system. However, software size can be expressed by many other measurements (Reference 8), including

1. Total physical lines of code (carriage returns)

2. Noncomment/nonblank physical lines of code

3. Executable lines of code (ELOC) (not including type declarations)

4. Statements (semicolons in Ada, which include type declarations)

Chart 5 describes the size of the Ada projects in the flight dynamics area using these four measurements. The FORTRAN project, GROSS, was also included in the summary for comparison. The GROSS project is the FORTRAN implementation of the GRODY project, and the GRODY/GROSS comparison has been detailed in previous papers. Because the GOESIM and UARSTELS projects are both telemetry simulators, they are also very similar in terms of their functionality. These two Ada projects are estimated to be between 75 and 78 thousand lines of code (KSLOC). In comparison, a typical telemetry simulator in FORTRAN consists of approximately 28 KSLOC.

Unless one counts only Ada statements, these figures tell us that the use of Ada results in many more lines of code than the use of FORTRAN. The increase in lines of code is not necessarily a

negative result. Rather, it is simply that the size of the system implemented in Ada will be larger than an equivalent system in FORTRAN. It is also clear that a precise definition is needed of what is a line of code in Ada and what code is included in that measurement.

Throughout the years of developing similar systems in FORTRAN in the flight dynamics area, the average level of software reuse has been between 15 and 20 percent (Reference 9). FORTRAN projects that attained a 35 percent or higher level of reuse of previously developed code are rare. After the first Ada project and with only 5 to 6 years of maturing in the environment, Ada projects have now achieved a software reuse rate of over 30 percent. This is already greater than the typical FORTRAN project. The UARSTELS project is expected to consist of more than 40 percent reused code. This trend of increasing software reuse is very promising.

2.3 LIFE-CYCLE EFFORT DISTRIBUTION (Chart 6)

The GROSS project followed the typical FORTRAN life-cycle effort distribution (Reference 10). Specifically, a small amount (8 percent) of the total effort expended on the project was spent during the pre-design or requirements analysis phase of the project; 27 percent of the effort was spent during the design phase, 40 percent during the code implementation phase; and 25 percent during the system testing phase. For the Ada projects, significant changes to the life cycle have not yet been observed. However, the Ada life cycle is changing slightly with each project and may soon show a different life cycle than that expected for a FORTRAN project. The life cycles for the second and third Ada projects are shifting slightly to show more design

time required with less system test time.

As the Ada environment matures and the SEL learns more about Ada, the life cycle is expected to continue shifting in the direction that the early literature has reported (Reference 11): more time spent in the design phase and less time in the system test phase. FORTRAN projects could assume the reuse of the life cycle based on past experience. This life cycle cannot be automatically reused in Ada, and more study is needed to determine the duration and products of each phase of an Ada project.

With the current projects, the SEL has not observed significant changes to the life-cycle phases. However, effort by phase is time driven. The SEL also collects effort data by activity across all phases. With this data the amount of effort spent on such activities as design, coding, and testing is very different than the distribution of effort on activities for FORTRAN projects. Much more time is spent on design for the Ada projects, but more analysis is still needed in this area.

2.4 ADA COST/PRODUCTIVITY (Chart 7)

Discussions on Ada productivity are somewhat confusing because so many interpretations exist of software size measures in Ada. Depending on the measurement used and an individual's inclination, one could determine that Ada is either as good or not as good as FORTRAN. Using the total lines of delivered code as a measure, the first, second, and third Ada projects show an improving productivity over time, and they show a productivity greater than FORTRAN. However, considering only code statements (excluding all comments and continued lines of code), the results are different. An increasing productivity trend remains in the

Ada projects over time, but the Ada projects have not yet achieved the productivity level of FORTRAN projects.

Within the flight dynamics environment, many software components are reused on FORTRAN projects. Since no Ada components existed previously, the first Ada projects were, in fact, developing a greater percentage of their delivered code than the typical FORTRAN project. Based on a past study by the SEL and on experience with FORTRAN projects, personnel concluded that reused code costs around 20 percent of the cost of new code (ref 15). The cost of reused code lies in the effort needed to test, integrate, and document the reused code in the new system. Using this estimate, reusability can be factored into software size by estimating the amount of developed code. Because of the differences in cost of new and reused code, developed code is calculated as the amount of new code plus 20 percent of the reused code. With software reusability factored in, the productivity for developed statements on Ada projects is approximately the same as that for FORTRAN projects.

The trends in Ada productivity are very positive. Again, lines of code must be clearly defined when discussing productivity. Using total number of lines as the measurement of software size, Ada productivity was always greater than FORTRAN productivity. However, due to the greater number of lines of an Ada project compared to a similar FORTRAN project, this measure can be misleading.

2.5 USE OF ADA FEATURES (Chart 8)

It is difficult to tell whether a given project really used the Ada language to its fullest capacity. Different applications may or may not need all the features available in Ada. However, in an effort to achieve some measurement in the use of the features available in the Ada language, the SEL identified six Ada features to monitor: generic packages, type declarations, packages, tasks, compilable PDL, and exception handling. The SEL then examined the code to see how little or how much these features were used.

The numbers of packages and type declarations were normalized to the size of the system, and the number of generic packages was divided by the total number of packages in the system. As seen in chart 8, the use of four of these features has evolved over time: generic packages, type declarations, packages, and tasking. Compilable PDL and exception handling did not show any trends. Perhaps it is too early to see results in these areas.

The average size of packages (in SLOC) for the first Ada projects is much higher than the average size of packages for the second and third Ada projects. This is due to a difference in the structuring method between the first Ada projects and all subsequent Ada projects (Reference 4). The first Ada projects were designed with one package at the root of each subsystem, which led to a heavily nested structure. In addition, nesting of package specifications with package bodies was used to control package visibility. Current Ada projects are utilizing the view of subsystems described by Grady Booch (Reference 12) as an abstract design entity whose interface is defined by a number of separately compilable packages, and nesting of Ada packages is limited to generic package instantiations.

The use of generic packages from the first to the current Ada projects seems to be increasing. More tham a third of the packages on current projects are generic packages. This higher use of generics reflects both a stronger emphasis on the development of verbatim reusable components and increased understanding of how to effectively utilize generic Ada packages within the flight dynamics area.

The use of strong typing within these software systems is also increasing, as measured by the number of type declarations per KSLOC. With experience, developers are more comfortable with the strong typing features of Ada and are using its capabilities to a fuller extent.

The use of tasking shows the most dramatic evolution over time for any particular Ada feature in the flight dynamics environment; its use has decreased markedly. The first Ada project, GRODY, contained eight tasks. However, from lessons learned on the GRODY project, personnel on subsequent Ada dynamics simulator projects have reduced that number to four tasks. Current telemetry simulator projects require no tasks at all. In the area of tasking, experience has shown that extensive use of this Ada feature is not appropriate for many applications. Although more extensive use of tasking might be very appropriate for other applications, the use of this Ada feature has definitely changed as project personnel have learned to use tasking only in those situations that are appropriate.

2.6 RELIABILITY, ERROR/CHANGE RATE AND CHARACTERISTICS (Charts 9 and 10)

The SEL measures software reliability by the number of changes or error corrections made to the software. For Ada projects, software error and change rates show a very positive trend. While it is too early to observe a definite difference from the FORTRAN rates, the reliability of the Ada projects is at least as good as that of FORTRAN projects. The error and change rates on the Ada projects are also declining over time, a promising trend. The types of errors also show an evolution from first through third Ada projects.

On a typical FORTRAN project, design errors amount to only 3 percent of the total errors on the project. For the first and second Ada projects, 25 to 35 percent of all errors were classified as design errors, a substantial increase. However, for the third Ada project, design errors are dropping significantly and are estimated to be approximately 7 percent. This rate is close to what is experienced on FORTRAN projects and clearly shows a maturation process with growing expertise in Ada.

Much of the literature on Ada reports that the use of Ada should help reduce the number of interface errors in the software (Reference 13). In our FORTRAN environment, about one-third of all errors on a project are interface errors. On our first and second Ada projects, the number of interface errors was not greatly reduced. Around one-fourth of the errors were interface errors. However, with current projects, the SEL is now seeing the expected results: interface errors are decreasing.

"Errors due to a previous change" is a category of errors that was caused by a previous modification to the software. The first Ada project showed a large jump in the number of these errors compared to those using FORTRAN. However, all subsequent Ada

projects show a rate for "errors due to a previous change" very similar to the FORTRAN rate. Many things probably contributed to this initial jump in the error rate: inexperience with Ada, inexperience with Ada design methodologies, and a nested software architecture that made the software much more complex. Again, the error profile is decreasing with the maturity of the Ada environment.

3.0 OVERALL OBSERVATIONS ON THE IMPACT OF ADA (Chart 11)

In summary, many aspects of software development with Ada have evolved as our Ada development environment has matured and our personnel have become more experienced in the use of Ada. The SEL has seen differences in the areas of cost, reliability, reuse, size, and use of Ada features.

A first Ada project can be expected to cost about 30 percent more than an equivalent FORTRAN project (Reference 14). However, the SEL has observed significant improvements over time as a development environment progresses to second and third uses of Ada.

The reliability of Ada projects is initially similar to what is expected in a mature FORTRAN environment. However, with time, one can expect to gain improvements as experience with the language increases.

Reuse is one of the most promising aspects of Ada. The proportion of reusable Ada software on our Ada projects exceeds the proportion of reusable FORTRAN software on our FORTRAN projects. This result was noted fairly early in our Ada projects, and our experience shows an increasing trend over time.

The size of an Ada system will be larger than a similar system in FORTRAN when considering SLOC. Size measurements can be misleading because different measurements reveal different results. Ratios of Ada to FORTRAN range from 3 to 1 for total physical lines to 1 to 1 for statements.

The use of Ada features definitely evolves with experience. As more experience is gained, some Ada features may be found to be inappropriate for specific applications. However, the lessons learned on an earlier project play an invaluable part in the success of later projects.

REFERENCES

1. Software Engineering Laboratory (SEL), SEL-85-002, Ada Training Evaluation and Recommendations, R. Murphy and M. Stark, October 1985

2. F. McGarry and R. Nelson, "An Experiment with Ada--The GRO Dynamics Simulator," NASA/GSFC, April 1985

3. SEL, SEL-81-104, The Software Engineering Laboratory, D. Card, F. McGarry, G. Page, et al., February 1982

4. --, SEL-88-003, Evolution of Ada Technology in the Flight Dynamics Area: Design Phase Analysis, K. Quimby and L. Esker, 1988

5. --, SEL-88-001, System Testing of a Production Ada Project: The GRODY Study, J. Seigle, L. Esker, and Y. Shi, November 1988

6. C. Brophy, S. Godfrey, et al., "Lessons Learned in the Implementation Phase of a Large Ada Project," Proceedings of the Sixth National Conference on Ada Technology, 1988.

7. SEL, SEL-87-004, Assessing the Ada Design Process and Its Implications: A Case Study, S. Godfrey and C. Brophy, July 1987

8. D. Firesmith, "Mixing Applies and Oranges: Or What Is an Ada Line of Code Anyway?," Ada Letters, September/October 1988

9. Computer Sciences Corporation (CSC), IM-88/083(59 253), Software Reuse Profile Study of Recent FORTRAN Projects in the Flight Dynamics Area, L. Esker, January 1989

10. SEL, SEL-81-205, Recommended Approach to Software Development, F. McGarry, G. Page, et al., April 1983

11. V. Castor and D. Preston, "Programmers Produce More With Ada," Defence Electronics, June 1987

12. G. Booch, Software Engineering With Ada. Menlo Park, CA: Benjamin/Cummings Publishing Company, 1983

13. The MITRE Corporation, Use of Ada for FAA's Advanced Automation System (AAS), V. Basili et al., April 1987

14. B. Boehm, "Improving Software Productivity," Computer, September 1987

15. SEL, SEL-84-001, Manager's Handbook For Software Development, W. Agresti, F. McGarry, et al., April 1984

THE VIEWGRAPH MATERIALS

FOR THE

F. MCGARRY PRESENTATION FOLLOW

EVOLVING IMPACTS OF Ada ON A PRODUCTION SOFTWARE ENVIRONMENT

FRANK McGARRY

NASA/GSFC

LINDA ESKER
KELVIN QUIMBY

COMPUTER SCIENCES CORPORATION

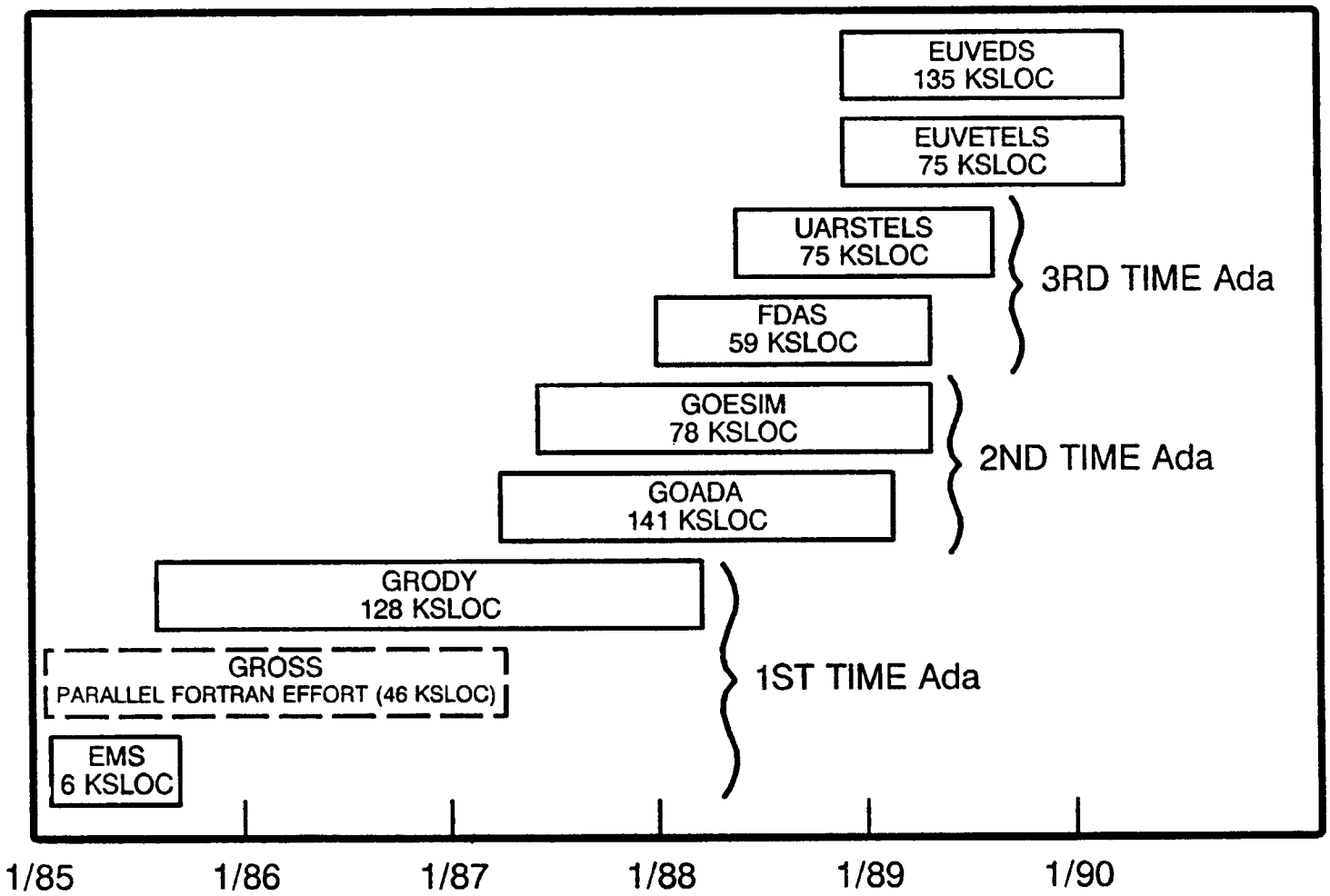
BACKGROUND

- Ada HAS BEEN UTILIZED ON 8 PROJECTS IN FLIGHT DYNAMICS DIVISION AT NASA/GSFC
- DETAILED DATA HAS BEEN COLLECTED/STUDIED FOR ALL PROJECTS - BY THE SEL
- APPROACH TO TRAINING/DESIGN/IMPLEMENTATION/TESTING HAS EVLOVED

POINTS TO BE ADDRESSED

- WHAT IS THE IMPACT OF Ada ON DEVELOPMENT PROFILES (E.G. EFFORT DISTRIBUTION)?
- WHAT ARE EFFECTS IN PRODUCTIVITY/RELIABILITY/MAINTAINABILITY...?
- WHAT IS THE CHANGING IMPACT OF Ada - 1ST TIME, 2ND TIME, 3RD TIME?
- DO WE USE Ada "DIFFERENTLY" OVER TIME...(BETTER %)?
- HOW LONG DOES IT TAKE TO REAP PROMISED BENEFITS?

Ada PROJECTS IN FLIGHT DYNAMICS DIVISION



F. McGarry
NASA/GSFC

D217.003

Ada PROJECTS STUDIED

PROJECT	TYPE	SIZE* (SLOC)	START DATE	DURATION	(11/30/88) STATUS	STAFF LEVEL
EMS	ELECTRONIC MAIL (PRACTICE/TRAINING)	5730	3/85	4 MO.	COMPLETE	7
GRODY	SIMULATOR (FLIGHT CONTROL SYSTEM)	128000	8/85	36 MO.	COMPLETE	7
GOADA	SIMULATOR (FLIGHT CONTROL SYSTEM)	141000	7/87	20 MO.	SYSTEM TEST	7
GOESIM	SIMULATOR (TELEMETRY)	78000	9/87	18 MO.	SYSTEM TEST	4
FDAS	EXECUTIVE (SOURCE CONTROL)	58700	1/88	13 MO.	SYSTEM TEST	4
UARSTELS	SIMULATOR (TELEMETRY)	75000	2/88	18 MO.	CODE	3

*SLOC = TOTAL LINES (CARRIAGE RETURNS) INCLUDES COMMENTS/BLANKS/REUSED
ALL PROJECTS DEVELOPED ON DEC VAX 11/780 OR VAX 8600

TEAM EXPERIENCE AND ENVIRONMENT USED

PROJECT	APPLICATION EXPERIENCE (RATIO)	AVERAGE S/W EXPERIENCE (YEARS)	Ada EXPERIENCE (RATIO)	ENVIRONMENT** COMPILABLE PDL	LSE	SD	CMS
EMS	NA	4.7	0	N	N	N	N
GRODY	1/7	4.7	0*	N	N	Y	Y
GOADA	2/7	5.9	3/7	Y	Y	Y	Y
GOESIM	1/4	5.7	1/4	Y	Y	Y	Y
FDAS	0/4	4.4	2/4	N	Y	Y	Y
UARSTELS	3/3	5.5	1/3	Y	Y	Y	Y

*TEAM HAD DEVELOPED SMALL Ada PRACTICE PROBLEM
**ALL DEVELOPMENT USED DEC.

SOFTWARE CHARACTERISTICS

	GROSS (FORTRAN)	GRODY (1ST TIME Ada)	GOADA (2ND TIME Ada)	GOESIM (2ND TIME Ada)	FDAS (3RD TIME Ada)	VARSTELS (3RD TIME Ada)	TYPICAL TM SIMULATION FORTRAN
TOTAL LINES (CR)	45500	128000	139000	78000	58700	75000	28000
NON COMMENT/ NON BLANK	26000	60000	68500	36000	31300	-	15000
EXECUTABLE LINES (NO TYPE DECL)	22500	40250	42000	21000	17100	-	12500
STATEMENTS (SEMI-COLON INCLUDES TYPE DECL)	22300	22500	25000	14000	11000	-	12000
% REUSED	36%	0	38%	32%	NA	42%	15%

1. Ada RESULTS IN LARGER SYSTEM (SLOC)
2. REUSE TREND VERY POSITIVE
3. "LINE OF CODE" DEFINITION CRITICAL

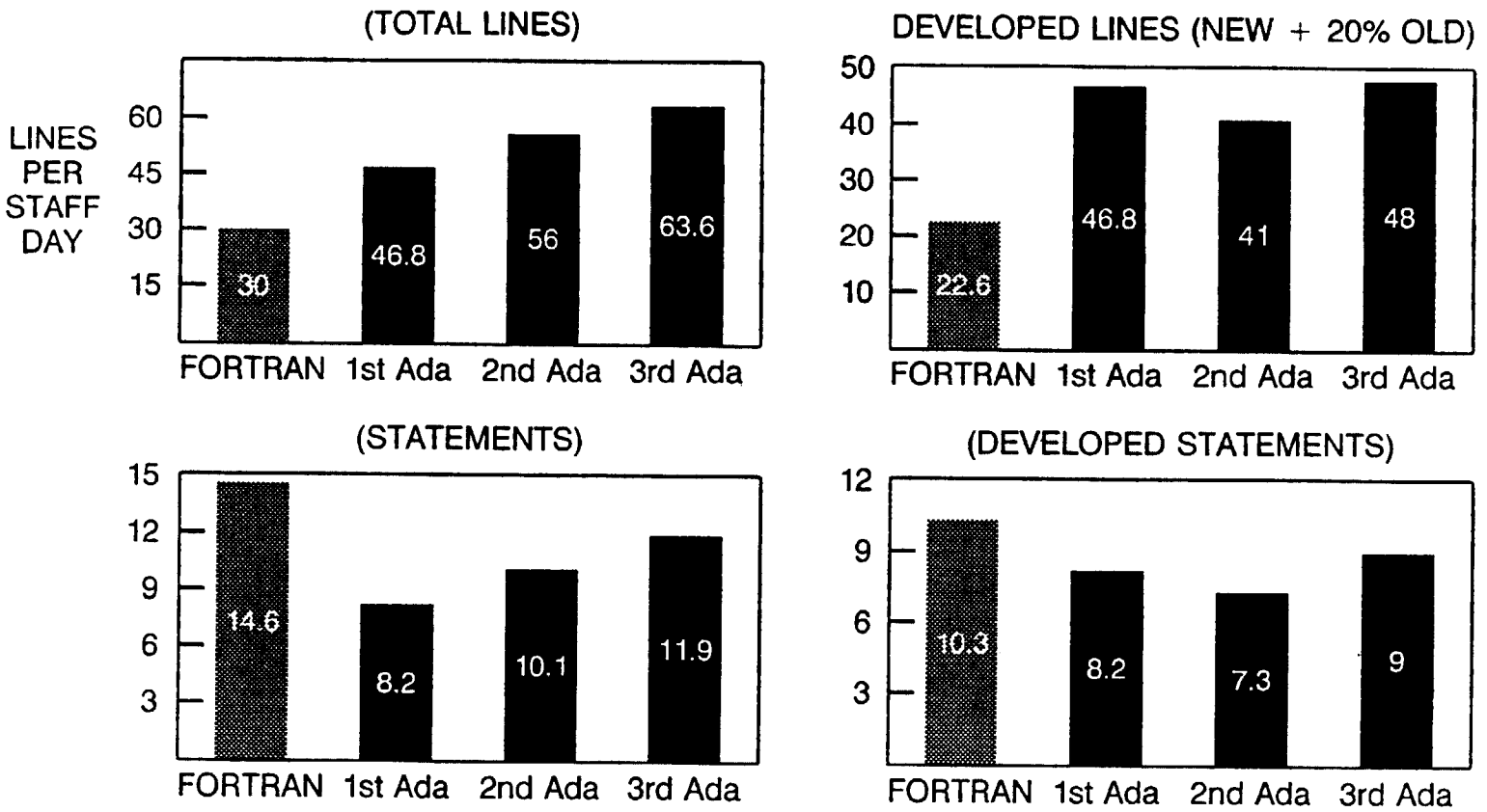
Ada IMPACTS ON LIFE CYCLE EFFORT DISTRIBUTION

	% TOTAL EFFORT*					
	GROSS (FORTRAN)	GRODY (1ST TIME Ada)	GOADA (2ND TIME Ada)	GOESIM (2ND TIME Ada)	FDAS (3RD TIME Ada)	UARSTELS (3RD TIME Ada)
PRE DESIGN	8	8	6	4	8	-
DESIGN	27	24	32	34	34	-
CODE	40	42	42	41	38	-
TEST	25	26	20	21	20	-
TOTAL EFFORT (HOURS)	12150	21860	21230**	10360**	7390**	-

SIGNIFICANT CHANGES TO LIFE CYCLE HAVE NOT YET BEEN OBSERVED - BUT . . .

*EFFORT DISTRIBUTION BASED ON PHASE DATES (E.G. END DESIGN, END CODE, END TEST)
** PARTIALLY BASED ON ESTIMATES TO COMPLETION

Ada COST/PRODUCTIVITY

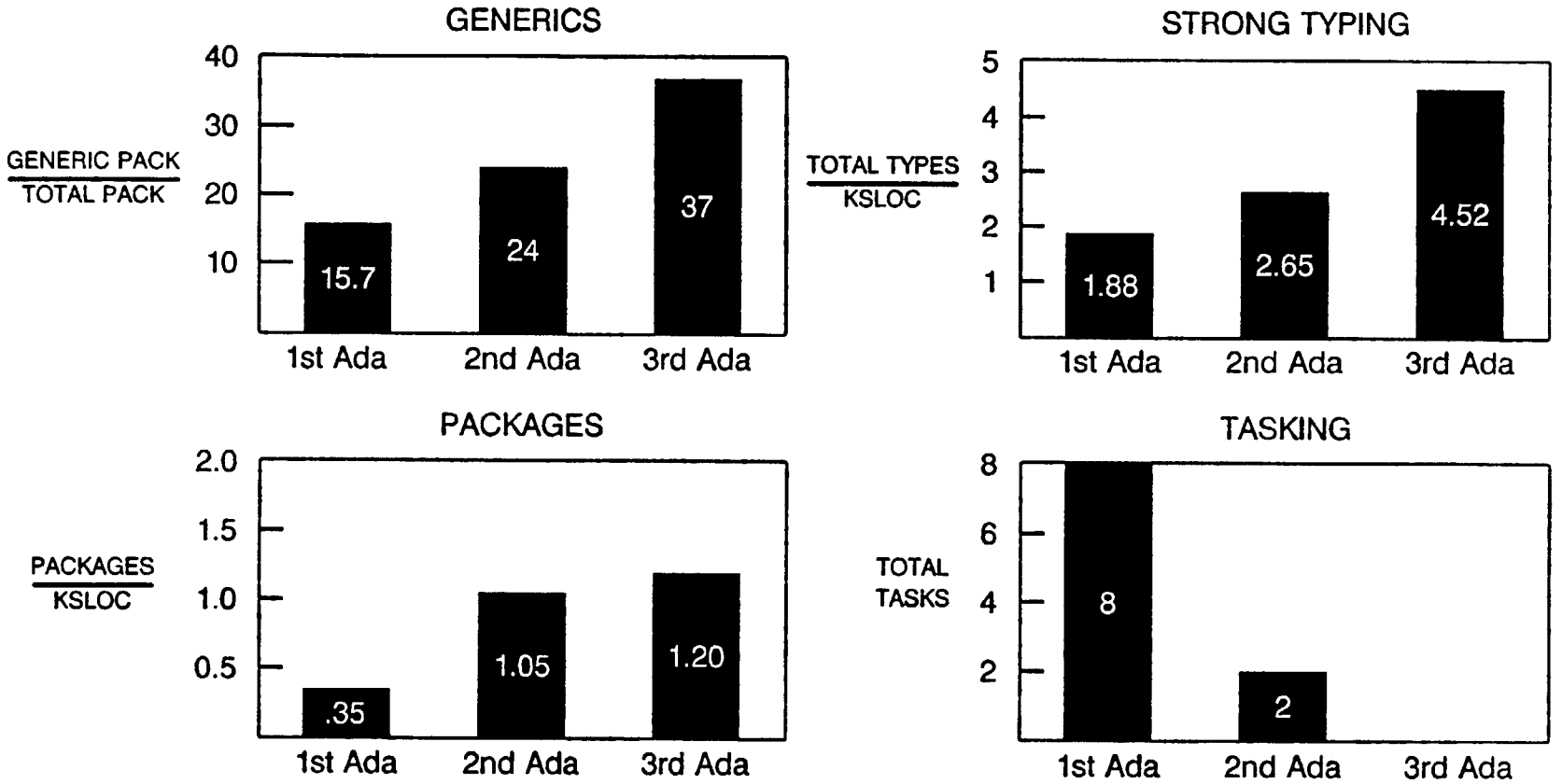


- CLEARLY DEFINE "LINES OF CODE"
- DO NOT USE SLOC IN COMPARING FORTRAN/Ada
- Ada TRENDS ARE IN POSITIVE DIRECTION

(GROSS/GRODY/GOADA/GOESIM/FDAS)

D217.016

USE OF Ada FEATURES



- USE OF Ada FEATURES CHANGES APPRECIATELY WITH EXPERIENCE
- NOT ALL FEATURE APPROPRIATE FOR APPLICATION

F. McGarry
NASA/GSFC

D217.018

Ada AND ERROR/CHANGE RATE

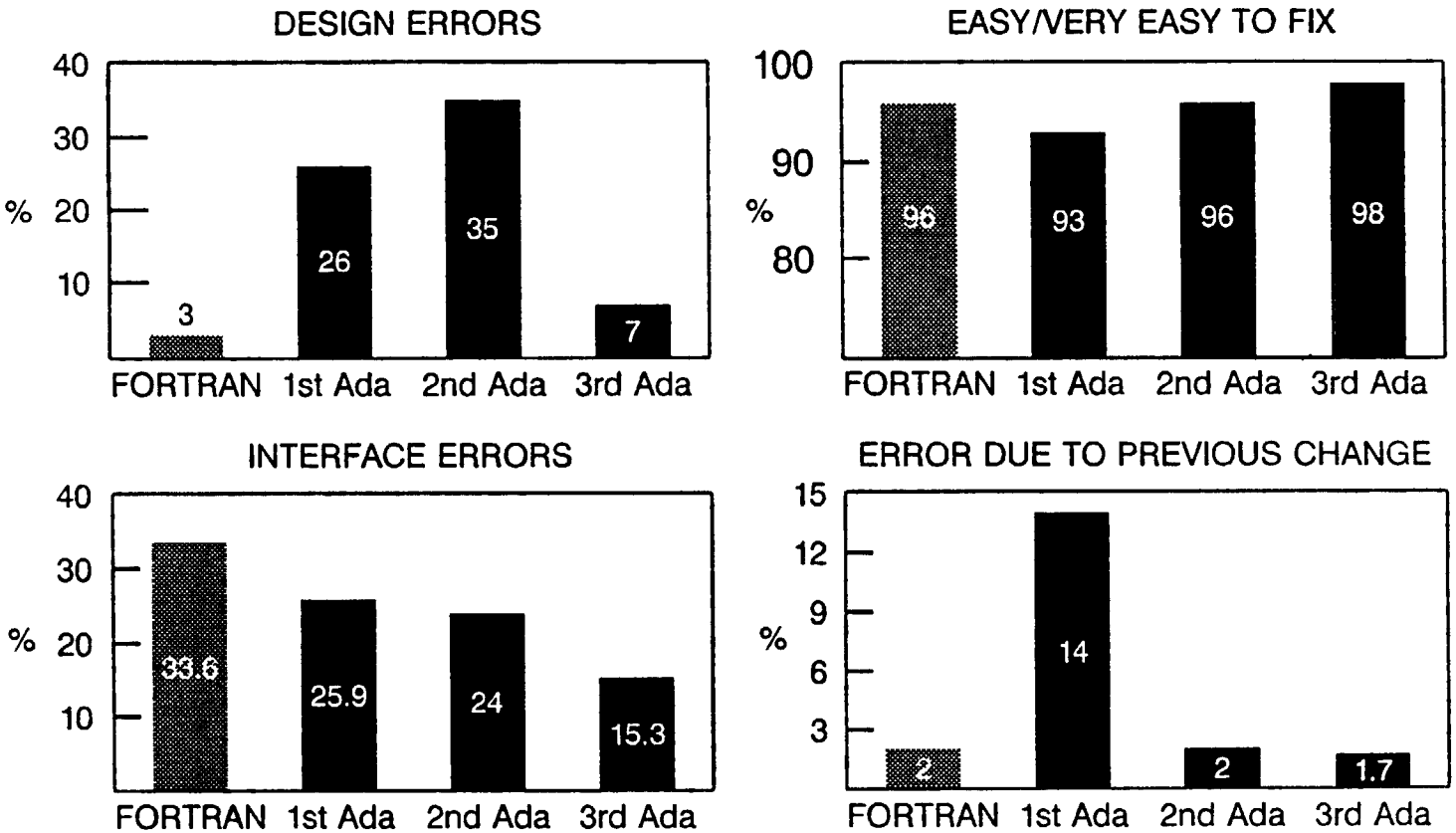
	GROSS (FORTRAN)	GRODY	GOADA	GOESIM	FDAS
CHANGES/KSLOC*	5.8	4.2	2.8	2.4	6.8
ERRORS/KSLOC	3.4	1.8	1.7	1.4	1.0

- RELIABILITY OF Ada SOFTWARE - AT LEAST AS GOOD AS FORTRAN
- VERY POSITIVE TRENDS FOR Ada - OVER TIME

*SLOC = TOTAL LINES (INCLUDES COMMENTS/REUSED)

D217.010

ERROR CHARACTERISTICS



- Ada ERROR PROFILE CHANGES WITH MATURITY OF USE
- Ada HELPS CUT INTERFACE ERRORS

EVOLVING IMPACTS OF Ada

OBSERVATIONS
(FROM 6 PROJECTS IN THE SEL)

1. COST
   - 30%+ OVERHEAD TO "FIRST TIME" PROJECTS
   - SIGNIFICANT IMPROVEMENTS ON 2ND TIME/3RD TIME USE

2. RELIABILITY
   - INITIALLY SIMILAR TO FORTRAN
   - IMPROVEMENTS WITH EXPERIENCE

3. REUSE
   - VERY POSITIVE TRENDS - EXCEEDS FORTRAN EXPERIENCE EARLY

4. SIZE

   (Ada TO FORTRAN - LARGER)

   - TOTAL LINES 3 TO 1
   - EXECUTABLE LINES 2 TO 1
   - NON COMMENTS 2 1/2 TO 1
   - STATEMENTS 1 TO 1

5. USE OF Ada FEATURE
   - PROMINANT EVOLUTION WITH "EXPERIENCE"
   - SEEMS RELATED TO IMPROVED DEVELOPMENT
   - SOME FEATURES INAPPROPRIATE TO SPECIFIC PROBLEMS

USE OF ADA FEATURES

	EMS*	GRODY	GOADA	GOESIM	FDAS*	UARSTELS
COMPILABLE PDL	N	N	Y	Y	N	Y
GENERICS [GENERIC PACKAGES / TOTAL PACKAGES]	0	15.7%	22%	25.8%	9%	37%
STRONG TYPING [TOTAL TYPES DECLARED / KSLOC]	3.86	1.88	2.37	2.96	1.41	4.52
PACKAGES [PACKAGES / KSLOC]	1.05	.35	.95	1.14	1.00	1.20
TASKING	0	8	3	0	1	0
EXCEPTION HANDLING [EXCEPTIONS / UNITS]	.41	.15	.24	.17	.10	.15

- USE OF ADA FEATURES CHANGES APPRECIABLY W/"EXPERIENCE"
- NOT ALL FEATURES APPROPRIATE FOR SPECIFIC APPLICATION
- LESSONS LEARNED' ANALYSIS EFFECTIVE (eg. TASKING)

*MUCH DIFFERENT APPLICATION THAN OTHERS

ERROR CHARACTERISTICS

	GROSS (FORTRAN)	(1ST TIME Ada)	(2ND TIME Ada)	(3RD TIME Ada)
% DESIGN ERRORS	3	25.9	35	7
% "EASY TO FIX + VERY EASY TO FIX"	96	93	96	98
% INTERFACE ERRORS	33.6	25.9	24	15.3
% ERRORS DUE TO PREVIOUS CHANGE	2	14	2	1.7

- Ada ERROR PROFILES CHANGING WITH MATURITY OF "ENVIRONMENT"
- Ada HELPS CUT INTERFACE ERRORS
- FORTRAN DESIGN REUSE APPARENT

Ada COST/PRODUCTIVITY

PRODUCTIVITY
(LINES PER STAFF-DAY)

	GROSS (FORTRAN)	GRODY	GOADA	GOESIM	FDAS
TOTAL LINES	30.0	46.8	52.4	60.2	63.6
DEVELOPED LINES (NEW +20% OLD)	22.6	46.8	37.0	45.5	47.8
NON COMMENT	17.1	22.0	25.8	27.8	33.9
STATEMENTS	14.7	8.2	9.4	10.8	11.9
DEVELOPMENT STMTS	10.3	8.2	6.9	8.3	9.0

- CLEARLY DEFINE "LINE OF CODE"
- DO NOT USE SLOC IN COMPARING FORTRAN/Ada
- Ada TRENDS ARE IN POSITIVE DIRECTION